BASIC STAMP

MANUEL D'UTILISATION

Référence : 2748

INTRODUCTION

Merci d'avoir acheté le BASIC Stamp. De prime abord, le Stamp vous semble peut-être très simple et, de fait il l'est. Mais, avec un logiciel qui utilise à plein ses capacités matérielles, le Stamp peut vous offrir des possibilités très intéressantes sans devoir investir dans du matériel coûteux.

Caractéristiques physigues
Le Stamp est constitué de deux composants principaux. Le cerveau du Stamp est un **microcontrôleur PIC 16C56** programmé avec notre interpréteur **PBASIC**. Il est associé à une **EEPROM de 256 octets** qui contient une version condensée de votre propre programme BASIC, lu et exécuté par l'interpréteur.

Le reste du Stamp est constitué :

- d'un résonateur céramique à 4 MHz
- d'un régulateur 5 volts
- de deux clips pour une pile de 9 volts.

Une petite zone de câblage rapide est également prévue sur le circuit imprimé avec des points de connexion pour les 8 lignes d'entrées/sorties du Stamp, l'alimentation 5 volts, l'alimentation non stabilisée et la masse.

Programmation
Le Stamp se programme avec notre langage BASIC simplifié. Ce langage utilise des instructions classiques telles que: FOR, NEXT, IF, THEN et GOTO ainsi que des instructions spécifiques adaptées au Stamp telles que: SERIN, PWM et BUTTON par exemple.
Chaque instruction utilise environ 2 ou 3 octets en mémoire EEPROM ce qui vous permet d'écrire de 80 à 100 instructions environ. La vitesse d'exécution quant à elle est de l'ordre de 2000 instructions par seconde.

Le kit de développement
Afin d'écrire des programmes pour le Stamp vous avez besoin d'un kit de développement. Ce kit contient notre éditeur pour compatible PC, un câble de connexion du PC au Stamp, un Stamp et ce manuel. En connectant le Stamp au PC avec le câble fourni, vous pouvez écrire vos programmes avec notre éditeur, puis les télécharger dans la mémoire du Stamp pour les faire exécuter.
Les circuits programmés en PBASIC.

Certains clients ont exprimé leur intérêt pour intégrer dans leurs applications des PIC 16C56 programmés avec notre interpréteur PBASIC. Ces circuits sont donc disponibles à la vente, à !'unité ou en quantité. Les différents composants associés : résonateur 4 MHz et mémoire EEPROM sont également disponibles pour ceux d'entre vous qui souhaitent réaliser leur propre circuit imprimé.

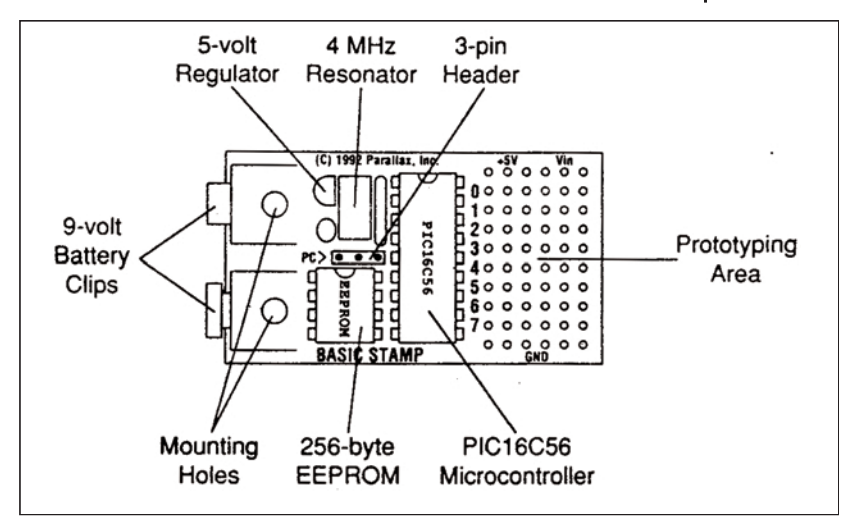
Pour finir :
La suite de ce manuel explique comment raccorder le Stamp au PC, comment utiliser l'éditeur et le logiciel de téléchargement et quelles sont les fonctions et syntaxes des diverses instructions du BASIC du Stamp.

SOMMAIRE

PAGE 5

DESCRIPTION MATERIELLE

Le schéma ci-dessous vous montre les différents constituants du Stamp :



Microcontrôleur PIC 16C56 : le cerveau du Stamp. Le microcontrôleur est programmé avec notre interpréteur PBASIC. Il lit vos programmes contenus en EEPROM et les exécute.

EEPROM : elle contient vos programmes sous une forme condensée et dispose de place supplémentaire pour vos données. Sa capacité totale est de 256 octets.

Résonateur 4 MHz : il pilote l'horloge du microcontrôleur.

Clips pour pile 9 volts : ils vous permettent d'alimenter directement le Stamp avec une simple pile de 9 volts standard (type 6F22). La consommation du Stamp est de 2 mA en mode normal et de 20µA en mode sommeil ; de ce fait le Stamp peut fonctionner longtemps avec une simple pile.

Régulateur 5 volts : il stabilise la tension d'alimentation du Stamp à 5 volts.

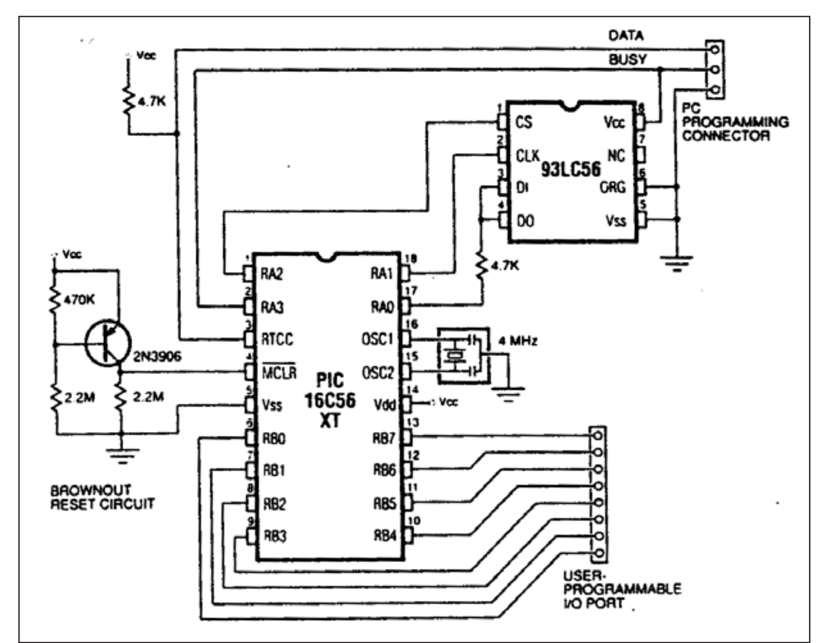
Connecteur à 3 points : destiné au câble de raccordement avec le compatible PC. Le câble doit être connecté lorsque vous voulez charger ou modifier un programme contenu dans l'EEPROM.

Zone prototype : elle vous permet de câbler quelques composants pour expérimenter vos circuits d'interface et permet également le raccordement avec les entrées/sorties du Stamp.

Trous de fixation : ils vous permettent de fixer fermement le Stamp dans votre application.

PAGE 6

Le schéma complet du Stamp vous est présenté ci-dessous.



Le Stamp utilise une résistance de 10 kΩ pour décharger le condensateur de 10µF de découplage de l'alimentation assurant ainsi un fonctionnement correct du circuit de reset automatique à la mise sous tension. Cette méthode donne satisfaction mais augmente la consommation du Stamp.
De plus, lors d'un arrêt, il est nécessaire de laisser le Stamp hors tension au moins 0,25 seconde avant de le remettre en marche.

La figure ci-dessus montre une circuiterie de reset à la mise sous tension qui utilise un transistor et trois résistances. Elle est plus efficace que la solution simple présentée ci-avant et équipera peu à peu tous les Stamp commercialisés. De plus, elle réagit à toute baisse anormale de la tension d'alimentation en provoquant un reset si nécessaire. Si vous intégrez des PIC 16C56 programmés en PBASIC dans vos applications, nous vous conseillons vivement l'utilisation de ce schéma.

PAGE 7

QUESTIONS / REPONSES CLASSIQUES

Quelle est la plage de tension d'alimentation utilisable pour le Stamp ?
Nous recommandons d'utiliser une simple pile de 9 volts car c'est une solution simple qui permet au Stamp de fonctionner de nombreux jours et même plusieurs semaines s'il passe en mode sommeil.

- Cependant si vous voulez utiliser une alimentation externe, vous pouvez utiliser n'importe quelle source délivrant de 3 à 12 volts sous un courant de 2 mA (hors consommation de vos circuits d'entrées/sorties).

- Si vous utilisez une source qui délivre de 3 à 5 volts, reliez-là à la borne + 5 volts du Stamp. Cela alimentera directement le Stamp sans passer par son propre régulateur.

- Si vous utilisez une source qui délivre de 6 à 12 volts, utilisez l'entrée Vin du Stamp afin de passer par le régulateur 5 volts du Stamp.

Puis-je utiliser le Stamp pour alimenter d'autres circuits?
Oui. Si vous avez besoin de 5 volts, utilisez la borne +5 volts du Stamp mais veillez à ne pas dépasser les possibilités du régulateur du Stamp (50 mA). Si vous avez besoin d'une tension non stabilisée (6 à 12 volts), utilisez la borne Vin.

Combien de temps peut fonctionner le Stamp sur une pile de 9 volts?
Cela dépend de l'utilisation que vous en faites. Si votre programme n'utilise jamais le mode sommeil et pilote de nombreuses LED, la durée de vie de la pile pourra être de quelques heures seulement. Si le mode sommeil est utilisé et si vos entrées/sorties consomment peu, le Stamp peut fonctionner plusieurs semaines sur sa pile.

Quel courant peuvent fournir ou absorber les entrées/sorties du Stamp ?
Chaque patte d'entrée/sortie du Stamp peut fournir 20 mA ou absorber 25 mA. Toutefois, le courant total fourni pour les 8 lignes ne doit pas excéder 40 mA et le courant total absorbé par ces mêmes 8 lignes ne doit pas dépasser 50 mA. Ces limites sont tout simplement celles du PIC 16C56 qui équipe le Stamp.

PAGE 8

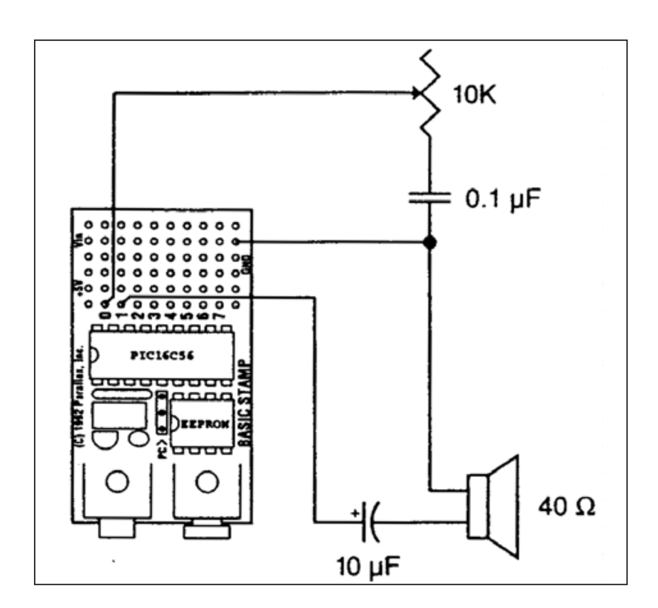
Le BASIC du Stamp supporte-t-il l'arithmétique en virgule flottante?
Non. Le Stamp travaille uniquement sur des entiers ce qui signifie qu'aucune valeur fractionnaire n'est autorisée. Les expressions doivent être fournies sous forme d'entiers et les résultats sont également des entiers. Ainsi, si vous tentez de diviser 5 par 2, le résultat sera 2.

Comment le Stamp évalue-t-il les expressions arithmétiques?
Les expressions sont évaluées strictement de la gauche vers la droite. Ceci peut parfois conduire à des erreurs et il importe donc d'être vigilant. Ainsi, si vous écrivez : 2 + 3 x 4, en arithmétique classique vous obtiendrez 14 comme résultat puisque la multiplication sera prioritaire. Avec le Stamp vous obtiendrez 20, exactement comme si vous aviez écrit (2 + 3) x 4.

PAGE 9

EXEMPLE D'APPLICATION

Cette page vous montre un exemple d'application très simple du Stamp. Son but est de lire la valeur d'un potentiomètre et de générer un signal de fréquence correspondante sur le haut-parleur.
Si la valeur du potentiomètre change, la fréquence du signal également. Le potentiomètre peut évidemment être remplacé par tout composant présentant une résistance variable, une LDR ou une CTN par exemple.



boucle :	pot 0,1 00,b2	'Lit le potentiomètre sur l'entrée 0 et place le résultat dans la variable b2
	b2 = b2/2	'Divise la lecture par 2 pour limiter sa valeur à 128
	sound 1 ,(b2, 10)	'Génère un signal sonore sur la sortie 1. La fréquence est la valeur de b2 et la durée du signal est fixée par 10
	goto loop	'Répète le processus indéfiniment.

PAGE IO

CONFIGURATION REQUISE

Pour pouvoir utiliser le kit de développement, il vous faut disposer d'un compatible PC dont voici la configuration minimum :

- Lecteur de disquette - Port parallèle
- 128 K de RAM
- MS DOS 2.0 ou supérieur.

Pour alimenter le Stamp, nous vous recommandons d'utiliser une simple pile de 9 volts. Si vous voulez employer une alimentation externe, vous pouvez utiliser n'importe quelle source délivrant de 3 à 12 volts sous un courant de 2 mA (hors consommation de vos circuits d'entrées/sorties).

- Si vous utilisez une source qui délivre de 3 à 5 volts, reliez-là à la borne + 5 volts du Stamp. Cela alimentera directement le Stamp sans passer par son propre régulateur.

• Si vous utilisez une source qui délivre de 6 à 12 volts, employez l'entrée Vin afin de passer par le régulateur 5 volts du Stamp.

Attention!
Si vous connectez une source de tension supérieure à 5 volts à la borne +5 volts du Stamp, vous pouvez l'endommager irrémédiablement.

PAGE 11

CONNEXION AU PC

Pour programmer le Stamp vous devez le connecter au PC et exécuter l'éditeur et le logiciel de téléchargement. Pour relier le Stamp au PC, procédez de la façon suivante :

-1- Avec votre kit de développement vous avez du recevoir un cordon muni à une extrémité d'un connecteur à 3 points et à l'autre d'une prise DB 25. Connectez la prise DB 25 sur le port **PARALLELE** de votre compatible PC.

-2- Connectez l'autre extrémité du câble sur le connecteur à 3 points du Stamp.
Attention! Veillez à bien mettre face à face la flèche visible sur le circuit imprimé du Stamp à côté du mot PC avec celle se trouvant sur le connecteur du câble.

-3- Alimentez le Stamp avec l'une des méthodes décrites ci-avant.
Avec le Stamp raccordé et alimenté, vous pouvez alors lancer l'éditeur et le logiciel de téléchargement comme expliqué dans la suite de ce manuel.

PAGE 12

UTILISATION DE L'EDITEUR

Lancement de l'éditeur
Avec le Stamp connecté et alimenté, lancez l'éditeur en frappant la commande suivante depuis l'invite du DOS:
STAMP
5
Sous réserve d'être dans le bon répertoire. l'éditeur va se lancer après quelques secondes. L'écran est bleu sombre avec une ligne en haut qui rappelle le mode d'accès à diverses fonctions relatives aux fichiers et au téléchargement. Hormis cette ligne, tout l'écran est disponible pour entrer vos programmes BASIC.

Formatage des programmes
L'entrée des programmes est très souple et ne doit respecter que peu de contraintes. Il vous faut cependant connaître les quelques règles relatives aux notations des constantes, des étiquettes et des commentaires, décrites ci-après.

Constantes : elles peuvent être déclarées de quatre manières différentes: décimal, binaire, hexadécimal ou ASCII.
Les nombres hexadécimaux sont précédés du symbole dollar ($). Les nombres binaires du symbole pour cent (%) et les constantes ASCII sont comprises entre des doubles guillemets (" "). Si aucun de ces symboles n'est utilisé, les constantes sont présumées être en décimal.

Voici quelques exemples :

100	Décimal
$64	Hexadécimal
%0110110	Binaire .
"A"	A en ASCII (65 ou $41)
"Hello"	"H", "e", "l", "l", "o" en ASCII

PAGE 13

Etiquettes d'adresses : l'éditeur utilise des étiquettes pour repérer des adresses ou emplacements particuliers de votre programme. Cela diffère de certains BASIC classiques qui n'utilisent que des numéros de ligne mais c'est beaucoup plus souple d'emploi.

En principe, une étiquette est n'importe quelle combinaison de lettres, chiffres et souligné (_) mais le premier caractère d'une étiquette ne doit jamais être un chiffre. De même, les étiquettes ne doivent pas être des noms d'instructions (serin, goto, etc...)

Lors de sa première utilisation, une étiquette doit se terminer par le caractère deux points (:). Lors des appels ultérieurs dans le programme, ces deux points ne doivent pas être utilisés.

Voici quelques exemples :

```
boucle :     toggle 0          'change l'état de la patte 0
             for b0=1 to 10
             toggle 1          'change l'état de la patte 1 10 fois
             next
             goto boucle       'répète le processus
```

Etiquettes de données : l'éditeur peut aussi utiliser des étiquettes pour définir des données.
Les même règles de syntaxe que celles vues pour les étiquettes d'adresses sont valables mais les étiquettes de données ne se terminent jamais par deux points (:) et doivent être définies grâce à la directive "symbol".

Voici quelques exemples :

```
symbol start = 1               'définit deux étiquettes de constantes
symbol end = 10
symbol count = B0              'définit une étiquette de variable
boucle :       for count = start to end
               toggle 1
               next
```

PAGE 14

Commentaires : ils peuvent être ajoutés à votre programme pour le rendre plus lisible.
Les commentaires commencent par une apostrophe (') et se terminent à la fin de la ligne. Si vous le désirez, vous pouvez aussi utiliser la directive REM classique des autres BASIC.

Voici quelques exemples :

```
symbol relay = 3               'définit une étiquette de constante
REM Voici la boucle principale
```

Formatage général. :
L'éditeur est insensible aux majuscules et minuscules sauf lors du traitement et de la définition des chaînes de caractères telles que "Hello" par exemple.
Plusieurs instructions et étiquettes peuvent être placées sur la même ligne séparées par le caractères deux points (:).

```
dirs = 255 : for b2 = 0 to 100 : pins = b2 : next
```

PAGE 15

Opérateurs arithmétiques : Les opérateurs suivants peuvent être utilisés en BASIC Stamp :

+	addition
-	soustraction
*****	multiplication (fournit le mot de poids faible du résultat)
******	multiplication (fournit le mot de poids fort du résultat)
/	division (fournit le quotient)
//	division (fournit le reste)
MIN	maintient une variable supérieure ou égale à une valeur
MAX	maintient une variable inférieure ou égale à une valeur
&	ET logique
I	OU logique
^	OU exclusif logique
&/	NON ET logique
I/	NON OU logique
^/	NON OU exclusif logique.

Voici quelques exemples :

```
count = count + 1    'augmente count de 1
timer = timer * 2    'multiplie le contenu de timer par 2
b2 = b2 / 8          'divise b2 par 8
w3 = w3 & 255        'extrait l'octet de poids faible de w3
```

PAGE 16

ENTREE ET EDITION DES PROGRAMMES

Hormis les quelques règles vues ci-avant, vous pouvez éditer vos programmes comme bon vous semble. Nous avons essayé de rendre cet éditeur aussi intuitif d'emploi que possible : pour aller vers le haut appuyez sur la touche "flèche haut” du curseur ; pour mettre en surbrillance un caractère sur la droite, actionnez shift et la touche "flèche à droite", et ainsi de suite.

La majorité des fonctions de l'éditeur est simple à utiliser. De plus, en une seule action sur des touches vous pouvez obtenir les fonctions suivantes :

- Chargement et sauvegarde d'un programme.
- Déplacement du curseur d'un caractère, d'un mot, d'une ligne, d'un écran ou au début ou à la fin d'une ligne.
- Mise en surbrillance d'un caractère, d'un mot, d'une ligne, d'un écran, depuis le début ou jusqu'à la fin d'une ligne.
- Couper, copier et coller du texte mis en surbrillance.
- Rechercher et/ou remplacer du texte.

TOUCHES DE FONCTIONS DE L'EDITEUR

Vous trouverez ci-dessous la liste complète des touches de fonctions de l'éditeur avec leur effet.

Alt-R	Télécharge le programme et lance l'exécution
Alt-L	Charge le programme depuis le disque
Alt-S	Sauvegarde le programme sur disque
Alt-Q	Quitte l'éditeur et retourne au DOS
Entrée (Enter)	Entre l'information et descend d'une ligne
Tab	Même fonction qu'entrée
Flèche à gauche	Déplace d'un caractère vers la gauche
Flèche à droite	Déplace d'un caractère vers la droite
Flèche en haut	Déplace d'une ligne vers le haut
Flèche en bas	Déplace d'une ligne vers le bas

PAGE 17

Ctrl-Flèche gauche	Déplace d'un mot vers la gauche
Ctrl-Flèche droite	Déplace d'un mot vers la droite
Home	Déplace au début de la ligne
Fin (End)	Déplace à la fin de la ligne
Page haute	Déplace d'un écran vers le haut
Page basse	Déplace d'un écran vers le bas
Ctrl-Page haute	Déplace au début du fichier
Ctrl-Page basse	Déplace à la fin du fichier
Shift-Flèche gauche	Mise en surbrillance d'un caractère à gauche
Shift-Flèche droite	Mise en surbrillance d'un caractère à droite
Shift-Flèche haute	Mise en surbrillance d'une ligne vers le haut
Shift-Flèche basse	Mise en surbrillance d'une ligne vers le bas
Shift-Ctrl-Flèche gauche	Mise en surbrillance d'un mot à gauche
Shift-Ctrl-Flèche droite	Mise en surbrillance d'un mot à droite
Shift-Home	Mise en surbrillance jusqu'au début de la ligne
Shift-Fin	Mise en surbrillance jusqu'à la fin de la ligne
Shift-Page haute	Mise en surbrillance d'un écran vers le haut
Shift-Page basse	Mise en surbrillance d'un écran vers le bas
Shift-Ctrl-Page haute	Mise en surbrillance jusqu'au début de la ligne
Shift-Ctrl-Page basse	Mise en surbrillance jusqu'à la fin de la ligne
Shift-insertion	Mise en surbrillance du mot où est le curseur
ESC	Annule la surbrillance en cours
Retour arrière	Efface un caractère à gauche
Suppression	Efface le caractère sous le curseur
Shift-Retour arrière	Efface du caractère de gauche au début de la ligne
Shift-Suppression	Efface jusqu'à la fin de la ligne
Ctrl-Retour arrière	Efface la ligne
Alt-X	Coupe le texte mis en surbrillance
Alt-C.	Copie le texte mis en surbrillance
Alt-V	Colle le texte coupé ou copié à l'endroit du curseur
Alt-F	Recherche une chaîne de caractères
Alt-N	Passe à l'apparition suivante de la chaîne
Alt-P	Calibre l'échelle du potentiomètre (voir inst. POT).

PAGE 18

EXECUTION DES PROGRAMMES

Pour exécuter le programme visible à l'écran, frappez Alt-R. L'éditeur va tester tous les ports parallèles du PC à la recherche du Stamp et, lorsqu'il l'aura trouvé, il va télécharger ce programme dans sa mémoire et lancer son exécution. Si l'éditeur ne trouve pas le Stamp, un message d'erreur est affiché.

- Si votre Stamp est correctement relié au PC, l'éditeur va afficher un barregraphe indiquant l'évolution du téléchargement. Cette opération pouvant ne durer que quelques secondes, le barregraphe se remplit très rapidement.
- Si vous examinez ce barregraphe vous remarquerez qu'une partie se remplit de carrés blancs et l'autre de carrés rouges. Les carrés blancs représentent la mémoire encore disponible dans l'EEPROM du Stamp et les rouges ce qui est occupé par votre programme.
- Lorsque le téléchargement est terminé, l'exécution de votre programme est automatiquement lancée. Si vous avez utilisé la directive DEBUG (voir ci-après), les données sont affichées sur l'écran du PC au fur et à mesure de leur évolution.
- Pour effacer le graphe du téléchargement de ''écran et revenir au mode édition, frappez n'importe quelle touche.

CHARGEMENT DES PROGRAMMES
Pour charger un programme depuis un fichier disque, frappez Alt-L. Un boîte de dialogue apparaît et vous demande le nom du fichier désiré. Si vous frappez le nom d'un programme qui existe effectivement il est chargé, dans le cas contraire un message d'erreur est affiché. Si vous changez d'avis, frappez ESC à la place du nom demandé.

SAUVEGARDE DES PROGRAMMES
Pour sauvegarder un programme sur disque, frappez Alt-S. Un boîte de dialogue s'affiche demandant le nom que vous voulez donner au fichier. Après avoir donné ce nom l'éditeur sauvegarde votre programme.

PAGE 19

UTILISATION DU "COUPER-COLLER"

Comme pour la majorité des logiciels actuels, l'éditeur du Stamp supporte les fonctions de couper-coller. Ceci permet de faire facilement d'importantes modifications à votre programme ou bien encore de répéter facilement des sous programmes identiques.

Cette fonction s'applique au texte préalablement mis en surbrillance et utilise un presse-papier intermédiaire qui est une zone mémoire spéciale de l'éditeur prévue pour cela. Lors d'un ordre couper ou copier, le texte est copié dans le presse-papier. Lors d'un ordre coller, il est copié du presse-papier vers l'emplacement où se trouve le curseur.

Notez bien que couper du texte n'est pas équivalent à l'effacer car si, dans les deux cas, le texte est bien enlevé d'où il se trouve, l'effacement le détruit définitivement alors que la fonction couper le met dans le presse-papier.

Voici un exemple simple d'utilisation de ces commandes :
- Commencez par mettre **un texte en surbrillance**, disons du curseur à la fin de la ligne par exemple. Pour cela frappez Shift-Fin. Tout ce qui se trouve du curseur à la fin de la ligne doit passer en surbrillance.
- Ensuite frappez **Alt-X**. Le texte doit disparaître.
- Amenez alors le curseur à l'endroit désiré puis frappez **Alt-V**. Le texte doit réapparaître, poussant si nécessaire ce qui lui fait suite.

La première étape aurait pu être remplacée par Alt-C au lieu de Alt-X si nous avions voulu copier le texte sans l'enlever de son emplacement initial.

PAGE 20

UTILISATION DE LA RECHERCHE ET REMPLACEMENT

L'éditeur dispose d'une fonction vous permettant de rechercher du texte et de le remplacer si vous le souhaitez. Cette fonction s'avère très utile si vous décidez, par exemple, de changer le nom d'une variable. Avec cette commande cela ne prend que quelques secondes pour agir sur tout votre programme alors que ce serait très fastidieux dans le cas contraire.

Pour utiliser cette fonction frappez **Alt-F**. Une boîte de dialogue apparaît et vous demande la chaîne de caractères à recherche ainsi qu'une chaîne de remplacement optionnelle.
Voici comment procéder :
- Frappez la chaîne à rechercher. Si elle contient "Entrée" ou "Tab" vous pouvez les inclure en frappant "Ctrl-Entrée" ou "Ctrl- Tab".

- **Frappez la chaîne de remplacement si nécessaire.** Elle sera placée dans le presse-papier. Lors de chaque apparition de la chaîne recherchée, il vous sera alors demandé si vous voulez la remplacer ou pas. Si vous voulez juste faire une recherche sans remplacement, frappez "Entrée" à la place de la chaîne de remplacement.

- L'éditeur efface alors la boîte de dialogue et met en surbrillance la première apparition de la chaîne. 'Pour la remplacer frappez Alt-V (coller de la fonction couper- coller).
Pour passer à l'apparition suivante frappez **Alt-N.**

PAGE 21

PORTS D'ENTREES/SORTIES ET ESPACE MEMOIRE

Le BASIC Stamp dispose de 16 adresses de mémoire vive ou RAM destinées au stockage des variables. Deux octets sont utilisés pour les entrées/sorties (un pour les données et l'autre pour le sens de fonctionnement) ce qui laisse en fait 14 octets pour les variables.

Ces 14 octets peuvent être utilisés comme des variables de type mots W0 à W6 ou comme des variables de type octets B0 à B13. De plus, B0 et B1 ou bien encore W0 peuvent être utilisés comme des variables bits (Bit0 à Bit15).
Tout ceci peut être résumé grâce au tableau ci-dessous :

Mots	**Octets**	**Bits**	**Autres noms de variables bits admis**
Port	Pins	Pin0-Pin7	Pins.0-Pins. 7, Port.0-Port.7
	Dirs	Dir0-Dir7	Dirs.0-Dirs.7, Port.8-Port.15
W0	B0	Bit0-Bit7	B0.0-B0.7, W0.0-W0.7
	B1	Bit8-Bit15	B1.0-B1.7, W0.8-W0.15
W1	B2		
	B3		
W2	B4		
	B5		
W3	B6		
	B7		
W4	B8		
	B9		
W5	B10		
	B11		
W6	B12		
	B13		

Le BASIC du Stamp est très souple quant aux noms qu'il est possible de donner aux variables comme le montre ce tableau. En fonction de vos besoins vous pouvez en effet utiliser l'espace mémoire comme des mots, des octets ou des bits. Le plus souvent ce sera ce dernier mode qui sera retenu, lors de la configuration ou de la lecture d'une ligne d'entrée/sortie par exemple.

PAGE 22

Le mot d'entrée/sortie **Port** est constitué de deux octets **Pins** et **Dirs :**
Pins et Pin0-Pin7 correspondent aux pattes d'entrées/sorties du port. Lorsque ces variables sont lues cela revient à lire effectivement les lignes d'entrées correspondantes. Lorsque vous écrivez dans ces variables, l'écriture a d'abord lieu en RAM puis est transférée sur les lignes de sorties correspondantes avant l'instruction suivante.

Dirs et Dir0-Dir7 sont les bits de définition de direction des ports d'entrées/sorties. Une mise à "0" de l'un de ces bits place la ligne correspondante du port en entrée et un "1" la place en sortie.
La prise en compte de cette affectation de sens à lieu avant l'exécution de l'instruction suivante.

Lorsque vous écrirez vos propres programmes, vous utiliserez de préférence ces noms de variables pour accéder au port d'entrées/sorties.
Une bonne pratique consiste à définir dès le début du programme quelle ligne va être entrée et quelle ligne va être sortie. Ainsi, en écrivant: dirs = %00001111 vous allez mettre les bits 0 à 3 du port en sorties et les bits 4 à 7 en entrées.

Après cette définition vous pouvez lire ou écrire directement sur les pattes correspondantes du port. Ainsi : pins =7 mettra au niveau logique "1" les bits 0 à 2 tandis que b2 = pins lira l'état des 8 lignes du port et les placera dans la variable b2.

Les pattes du port peuvent être adressées bit par bit ce qui peut être plus pratique pour certaines applications. Ainsi : if pin3 = 1 then start lira la patte 3 du port d'entrées/sorties et ira à l'adresse start si elle est à "1".

Les autres variables W0 à W6 peuvent être utilisées comme bon vous semble à une exception près : **W6 est utilisée comme pile si un GOSUB est exécuté.**

L'éditeur du Stamp reconnaît automatiquement les noms de variables vus ci-avant.
Si vous souhaitez faire appel à des noms différents pour rendre votre programme plus lisible, il vous suffit de faire appel à la directive symbol comme dans cet exemple :

```
symbol inter = pin0          'définit la patte 0 du port comme s'appelant inter
symbol drapeau=bit0          'définit le nom d'un bit
symbol compte = b2           'définit le nom d'un octet
```

PAGE 23

RESUME DU JEU D'INSTRUCTIONS

BRANCHEMENTS

IF ... THEN	Comparaison et branchement conditionnel.
BRANCH	Branchement à une adresse spécifiée par un décalage.
GOTO	Branchement à une adresse.
GOSUB	Appel de sous programme (jusqu'à 16 au maximum).
RETURN	Retour de sous programme.

BOUCLES

FOR .. NEXT	Boucle FOR ... NEXT.

INSTRUCTIONS NUMERIQUES

(LET)	Instruction optionnelle. Définition d'une opération sur une variable par ex. LET B = 2.
LOOKUP	Cherche une donnée spécifiée par un décalage dans une liste et la place dans une variable.
LOOKDOWN	Cherche la présence d'une donnée dans une liste et indique sa position.
RANDOM	Génère un nombre pseudo aléatoire.

ENTREES/SORTIES NUMERIQUES

OUTPUT	Met une patte en sortie.
LOW	Met une patte au niveau bas.
HIGH	Met une patte au niveau haut.
TOGGLE	Met une patte en sortie et inverse son niveau.
PULSOUT	Génère une impulsion en inversant temporairement le niveau d'une patte.
INPUT	Met une patte en entrée.
PULSIN	Mesure la durée d'une impulsion en entrée.
REVERSE	Inverse la fonction d'une patte (sortie - entrée et vice versa).
BUTTON	Anti-rebondissement d'un poussoir, répétition automatique, branchement à une adresse s'il est en état stable.

PAGE 24

ENTREES/SORTIES SERIES

SERIN	Reçoit des caractères sur une liaison série avec de très nombreuses possibilités de paramétrage (voir description détaillée de cette instruction).
SEROUT	Emet des caractères sous forme série (idem SERIN).

ENTREES/SORTIES ANALOGIQUES

PWM	Génère un signal impulsionnel modulé en largeur puis remet la patte correspondante en entrée.
POT	Lit un potentiomètre de 5 à 50 kΩ et normalise le résultat.

SONS

SOUND	Joue des notes. Un 0 correspond à un silence tandis que les notes vont en montant de 1 à 127. 128 à 255 fait générer des bruits blancs.

ACCES A L'EEPROM

EEPROM	Stocke des données dans l'EEPROM avant d'y télécharger le programme BASIC.
READ	Lit un octet de l'EEPROM et le met dans une variable.
WRITE	Ecrit un octet dans l'EEPROM.

INSTRUCTIONS TEMPORELLES

PAUSE	Suspend l'exécution du programme de 0 à 65536 ms.

CONTROLE DU PROGRAMME

NAP	Mise en mode sommeil pendant un court instant. La consommation est réduite à 20 µA.
SLEEP	Mise en mode sommeil de 1 à 65535 secondes. La consommation est réduite à 20 µA.
END	Mise en mode sommeil jusqu'à détection d'une connexion avec le PC. La consommation est réduite à 20 µA.

PAGE 25

CORRECTION DU PROGRAMME

DEBUG	Envoie les valeurs des variables au PC en phase de mise au point de programme.

PAGE 26

INSTRUCTIONS EN BASIC

BRANCH

BRANCH décalage (adresse 0, adresse 1, , adresse N)

Branche à l'adresse spécifiée par le décalage.

- **Décalage** est une constante ou une variable qui précise quelle adresse va être utilisée (0 à N).
- **Adresse** X est une étiquette définissant une adresse.

Exemple :

```
branch b2, ( abc, def, ghi)     'va à l'adresse abc si b2 vaut 0
                                'va à l'adresse def si b2 vaut 1
                                'va à l'adresse ghi si b2 vaut 2 .
                                'continue en séquence si b2 est une autre valeur
```

PAGE 27

BUTTON

BUTTON patte, état bas, délai, vitesse, variable, état cible, adresse

Lit l'état d'un poussoir et effectue un anti-rebondissement, l'auto-répétition, le branchement si le poussoir est dans un état stable. Le poussoir peut-être actif haut ou actif bas comme schématisé ci-après.

- Patte indique la patte d'entrée/sortie à utiliser.

- Etatbas est une constante ou variable logique (0 ou 1) qui précise l'état logique du poussoir lorsqu'il est appuyé.

- Délai est une constante ou variable de 0 à 255 qui indique pendant combien de temps le poussoir doit être actionné pour générer un fonction de répétition automatique. Si délai vaut 0 les fonctions d'anti-rebondissement et de répétition automatique sont inhibées. Si délai vaut 255 l'anti-rebondissement a lieu mais pas fa répétition automatique.

- Vitesse est une constante ou variable de 0 à 255 qui précise la vitesse de la répétition automatique. - Cette vitesse est exprimé en nombre de cycle de l'instruction BUTTON.

- Variable est la variable de travail de BUTTON. Elle doit être mise à zéro avant la première utilisation de cette instruction.

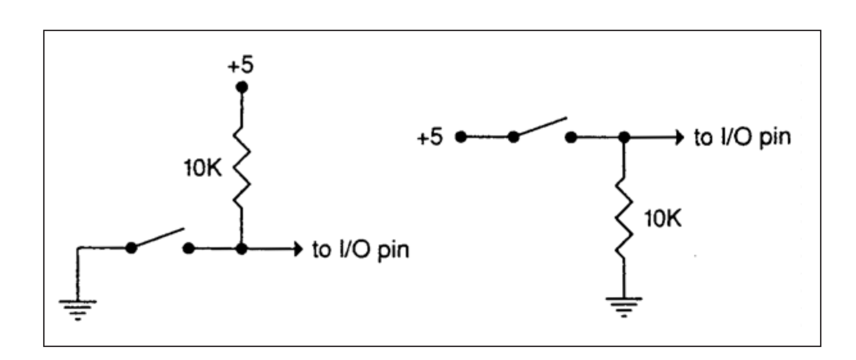
- Etatcible est une constante ou variable logique qui précise dans quel état doit être le poussoir pour que le branchement ait lieu (0 = non appuyé, 1 = appuyé).

- Adresse est une étiquette qui précise l'adresse où va brancher le programme lorsque le poussoir sera dans l'état définit par étatcible.

PAGE 28

Exemple :

b2 = 0	‘mise à 0 de la variable utilisée avant usage de BUTTON
button 3,0,90,10,b2,0,abc	'lecture du poussoir connecté sur la patte 3 (3)
	'patte au niveau bas lorsque le poussoir est appuyé (0)
	'répétition automatique après 50 cycles de la routine
	'vitesse de répétition 10 cycles de la routine
	'variable de travail utilisée b2
	'si le bouton n'est pas appuyé (0) saut à l'adresse abc



PAGE 29

DEBUG

DEBUG Envoie le contenu des variables au PC afin de les visualiser. Au point de vue formatage d'instruction, DEBUG fonctionne comme le PRINT de la majorité des BASIC mais, au lieu de faire imprimer ce qui suit, il le fait afficher sur l'écran du PC.

Les règles suivantes sont à considérer :

- Les variables identifiées par leurs noms feront afficher "variable = valeur' sur l'écran du PC.
- Les variables identifiées par leurs noms immédiatement précédés de # feront afficher seulement leur valeur, sans le texte "variable =".
- Les textes entre guillemets apparaîtront sur l'écran du PC comme ils sont frappés.
- Les variables sont normalement affichées en décimal. Pour les faire afficher en hexadécimal faites précéder leur nom du symbole dollar ($) et pour les faire afficher en binaire du symbole pour cent (%).
- cr peut être utilisé pour faire générer un retour chariot.
- cls peut être utilisé pour faire générer un effacement d'écran.
- les différents éléments à afficher doivent être séparés par des virgules.

Voici quelques exemples :

```
debug b2                          'affiche "b2 = valeur de b2"
debug #b2                         'affiche seulement la valeur de b2
debug "La donnée vaut ",b2        'affiche "La donnée vaut valeur de b2" .
debug #%b2                        'affiche seulement la valeur de b2 mais en binaire
debug "Entrées ",b2, b3,cr        'affiche "Entrées valeur de b2, valeur de b3" suivi d'un
                                   retour chariot.
```

PAGE 30

EEPROM

EEPROM {adresse},(donnée, donnée, ...)

Place des données dans l'EEPROM avant d'y charger le programme BASIC. Cette instruction est utile pour initialiser l'EEPROM avec des données qui seront ensuite utilisées par le programme.

Cette instruction est exécutée avant chaque téléchargement mais n'est pas elle même téléchargée en même temps que le programme. De ce fait elle ne consomme aucun espace mémoire.

- Adresse est une constante ou variable optionnelle comprise entre 0 et 255 qui indique à partir de quelle adresse les données doivent être stockées. Si elle est omise, le premier emplacement disponible est utilisé.

- Donnée est une constante ou une variable de 0 à 255 qui est placée dans l'EEPROM à partir de l'adresse spécifiée.

Exemple :

```
EEPROM 0,(5, 23,17)       'Ecrit 5 puis 23 puis 17 à partir de l'adresse 0 de l'EEPROM
```

PAGE 31

END

END Fait passer le Stamp en mode sommeil pour une durée indéfinie. Le retour au fonctionnement normal n'aura lieu que lors de la détection d'une demande de connexion de la part du PC.

La consommation du Stamp dans ce mode est de 20 µA (hors circuiterie d'entrée/sortie éventuelle). END n'utilise aucun paramètre.

PAGE 32

FOR ... NEXT
FOR variable = début TO fin {STEP {-} pas}
.
.
NEXT {variable}
Réalise une boucle FORT - NEXT classique. La variable est rendue égale à la valeur début puis le code compris entre FOR et NEXT est exécuté. La variable est alors incrémentée de 1 (ou du pas spécifié par la directive optionnelle STEP) et, si elle n'est pas égale ou supérieure à fin, les instructions comprises entre FOR et NEXT sont exécutées à nouveau et ainsi de suite. Lorsque la variable devient égale ou supérieure à fin, le programme continue en séquence après NEXT.

Quelles que soient les valeurs de début et de fin la boucle est toujours exécutée au moins une fois. Votre programme peut comporter autant de boucles FOR - NEXT que vous le souhaitez mais vous ne devez pas imbriquer les unes dans les autres plus de 8 boucles.

- Variable est une variable de type bit, octet ou mot utilisée comme compteur de boucle. Début et fin sont limités par la capacité en taille de cette variable (0 à 1, 0 à 255 ou 0 à 65535).

- Début est une constante ou une variable qui spécifie la valeur de début adoptée par la variable de boucle.

- Fin est une constante ou une variable qui spécifie la valeur de fin que devra atteindre ou dépasser (si l'égalité n'est pas possible) la variable de boucle.

- Pas est une donnée optionnelle et précise de combien doit être incrémentée ou décrémentée (si le signe - est présent) la variable de boucle. En son absence, le pas est fixé à + 1.

PAGE 33

Exemple :

```
	for b2 = 0 to 255	'définit une boucle de variable b2 qui sera exécutée 256 fois
	pins = b2		'fait sortir b2 sur le port d'entrée/sortie du Stamp
	next
```

PAGE 34

GOSUB
GOSUB adresse
Exécute un saut au sous programme dont l'adresse est spécifiée.
Lorsque le sous programme est terminé, l'instruction RETURN (voir ce mot) fait poursuivre l'exécution à l'instruction qui suit GOSUB.
Vous pouvez utiliser jusqu'à 16 GOSUB dans un seul et même programme.

- Adresse est une étiquette qui précise l'adresse de début du sous programme.

Exemple :

```
	forb4=o to10
	gosub abc		'exécute le sous programme abc
	next
abc: pulsout 0, b4		'génère une impulsion sur la patte 0 de durée égale à b4 x 10µs
	toggle 1		'change l'état de la patte 1
	return			'retour au programme appelant
```

PAGE 35

GOTO

GOTO adresse

Exécute un saut à l'adresse spécifiée et fait poursuivre l'exécution du programme à cette adresse.

- Adresse est une étiquette qui précise l'adresse où poursuivre l'exécution du programme.

Exemple :

```
abc : pulsout 0, 100          'génère une impulsion de 1 000µs sur la patte 0
      goto abc                'répète le processus
```

PAGE 36

HIGH

HIGH patte

Fait passer la patte spécifiée au niveau logique haut. Si la patte a été préalablement définie comme étant une entrée, elle est automatiquement mise en sortie par cette instruction.

- Patte est une constante ou variable (0 à 7) qui indique le numéro de patte à utiliser.

Exemple :

```
abc :  high 3                 'met la patte 3 au niveau haut
       low 2                  'met la patte 2 au niveau bas
       high 2                 'met la patte 2 au niveau haut
       low 3                  'met la patte 3 au niveau bas
       goto abc               'répète le processus
```

PAGE 37

IF ... THEN

IF variable ?? valeur {AND/OR variable ?? valeur ...} THEN adresse

Compare la ou les variables à la ou aux valeurs indiquées et effectue un branchement à l'adresse spécifiée si le résultat de la comparaison est vrai. Contrairement à ce qui est permis avec certains BASIC, l'adresse qui suit le THEN ne peut pas être remplacée par une instruction.
THEN doit impérativement être suivi par une adresse.

- ?? est un des opérateurs de relation suivants: =, <>, >, >, >=, <=.

- Variable est la variable qui sera comparée à "valeur'.

- Valeur est une constante ou une variable à la laquelle est comparée la variable.

- Adresse est une étiquette qui indique l'adresse à laquelle se continue l'exécution du programme si la comparaison est vraie.

Exemple :

```
abc:   serin 0,n2400,b2       'reçoit une donné série et la place dans b2
       if b2 = 185 then xyz   'si 185 est reçu, saut à xyz
       high 2                 'met la patte 2 au niveau haut
       goto abc               'répète le processus
xyz:   low 2                  'met la patte 2 au niveau bas
       goto abc               'répète le processus
```

PAGE 38

INPUT

INPUT patte

Place la patte correspondante en entrée ce qui permet ensuite à votre programme de lire le niveau logique auquel elle se trouve.

- **Patte** est une constante ou variable (0 à 7) qui indique le numéro de patte à utiliser.

Exemple :

```
        input 5                 'met la patte 5 en entrée
abc:    if pin5 = 1 then xyz    'si la patte 5 est au niveau haut, exécution de xyz
        high 2                  'sinon mise au niveau haut de la patte 2
        goto abc                'on répète le processus
xyz:    low 2                   'mise au niveau bas de la patte 2
        goto abc                'on répète le processus .
```

PAGE 39

LET

{LET} variable = {-} valeur ?? valeur ...

Assigne une valeur à une variable et/ou réalise diverses opérations arithmétiques et logiques sur une variable. Toutes ces opérations sont réalisées en interne au niveau d'un mot (16 bits).
L'instruction LET est optionnelle : LET a = 10 a donc la même signification que a = 10".

- **??** est un des opérateurs suivants :

+	addition
-	soustraction
*****	multiplication (fournit le mot de poids faible du résultat)
******	multiplication (fournit le mot de poids fort du résultat)
/	division (fournit le quotient)
//	division (fournit le reste)
MIN	maintient une variable supérieure ou égale à une valeur
MAX	maintient une variable inférieure ou égale à une valeur
&	ET logique
I	OU logique
^	OU exclusif logique
&/	NON ET logique
I/	NON OU logique
^/	NON OU exclusif logique

- **Variable** est le nom de la variable affectée par l'instruction.
- **Valeur** est une constante ou une variable utilisée par l'instruction.

Exemple :

```
        let b3 = b3 1 2         'divise b3 par 2
        b3 = b3 max 100         'limite b3 à la plage 0 - 100
```

PAGE 40

LOOKDOWN

LOOKDOWN cible, (valeur0, valeur1, ... valeurN), variable

Compare une liste de valeurs avec une donnée cible. Si une égalité est trouvée, place le numéro de rang de cette valeur (de 0 à N) dans la variable. Si aucune valeur ne concorde, la variable n'est pas affectée. Si par exemple la liste de valeurs est : 4, 13, 15, 28, 8 et que la cible est 15, la variable contiendra 2 puisque 15 est la troisième valeur de la liste et que le décompte commence à partir de 0.

- Cible est la constante ou la variable dont vous recherchez la présence dans une liste de valeurs.

- Valeur0 à ValeurN est la liste des valeurs auxquelles va être comparée la cible.

- Variable contient le résultat de la recherche ou n'est pas affectée si la recherche est négative.

Exemple :

```
serin 0,n2400,b2                'entrée de b2 par une liaison série
lookdown b2,(65,88,93),b3       ' si b2 = 65, b3 est rendu égal à 0
                                ‘ si b2 = 88, b3 est rendu égal à 1
                                ‘ si b2 -= 93, b3 est rendu égal à 2
```

PAGE 41

LOOKUP

LOOKUP décalage,(valeur0, valeur1, ..., valeurN),variable

Recherche la valeur spécifiée par décalage dans la liste de valeurs et place celle-ci dans variable. Si par exemple la liste de valeurs est 2, 13, 15, 28 et 8 et que le décalage vaille 1, la valeur placée dans la variable sera 13 puisque le décompte commence à partir de o. Si le décalage est plus grand que le nombre de valeurs disponibles, la variable n'est pas affectée.

- Décalage spécifie la position de la valeur de la liste qui doit être placée dans la variable.

- Valeur0 à ValeurN est la liste des valeurs disponibles.

- Variable contient le résultat de l'instruction.

Exemple :

```
for b2 = 0 to 25
lookup b2,(65, 66, 67, ...),b3  'convertit la valeur du décalage de 0 à 25 en la suite des
                                 caractères de A à Z placés dans b3
next
```

PAGE 42

LOW

LOW patte

Fait passer la patte spécifiée au niveau logique bas. Si la patte a été préalablement définie comme étant une entrée, elle est automatiquement mise en sortie par cette instruction.

- Patte est une constante ou variable (0 à 7) **qui indique le numéro de patte à utiliser**.

Exemple :

```
abc:    low 3                   'met la patte 3 au niveau bas
        high 2                  'met la patte 2 au niveau haut
        low 2                   'met la patte 2 au niveau bas
        high 3                  'met la patte 3 au niveau haut
        goto abc                'répète le processus
```

PAGE 43

NAP

NAP période

Place le Stamp en mode sommeil pour une courte durée. Dans ce mode la consommation est de 20µA hors consommation de la circuiterie d'entrées/sorties.

- Période est une constante ou variable qui spécifie la durée de ce mode selon la relation approximative suivante: 21^période * 18 en ms. Période peut varier de 0 à 7 ce qui donne une plage de durée de 18 ms à 2,3 secondes environ.

Exemple :

```
        nap 7                   'met le Stamp en mode sommeil pendant 2304 ms
```

PAGE 44

OUTPUT

OUTPUT patte

Place la patte correspondante en sortie.

- Patte est une constante ou variable (0 à 7) qui indique le numéro de patte à utiliser.

Exemple :

```
        bit0 = 0
        bit1 = 1
        output 2                'met la patte 2 en sortie
abc:    pin2 = bit0             'met la patte 2 au niveau bas
        pin2 = bit1             'met la patte 2 au niveau haut
        goto abc                'répète le processus
```

PAGE 45

PAUSE

PAUSE millisecondes

Suspend l'exécution du programme pendant un certain nombre de millisecondes. La précision de cette durée ne dépend que de celle du résonateur d'horloge du Stamp. Cependant il faut noter qu'un peu de temps (typiquement 1 ms) peut être perdu à exécuter les instructions adjacentes. L'erreur ainsi introduite est d'autant plus négligeable que la pause est longue.

Un timer relativement précis peut être réalisé grâce à cette instruction en effectuant des pauses de "longue" durée, telle que 100 ms par exemple, dans une boucle et en incrémentant un compteur à chaque fois. Tant que le compteur ne déborde pas, la boucle contenant la pause se répète.

- Millisecondes est une constante ou variable de 0 à 65535 qui spécifie la durée de la pause.

Exemple :

```
abc:    low 2                   'met la patte 2 au niveau bas
        pause 100               'pendant environ 100 ms
        high 2                  'met la patte 2 au niveau haut
        pause 100               'pendant environ 100 ms
        goto abc                'répète le processus
```

PAGE 46

POT

POT patte, échelle, variable

Lit un potentiomètre ou toute autre résistance variable (CTN, LOR, etc...) de valeur comprise entre 5 kohms et 50 kohms. La patte d'entrée doit être connectée à une extrémité de la résistance dont l'autre extrémité retourne à la masse par un condensateur de 0,1µF comme indiqué sur le schéma ci-après. La détermination de la résistance est faite par mesure du temps de charge du condensateur.

- Patte est une constante ou variable de 0 à 7 qui indique la patte à utiliser.

- Echelle est une constante ou variable de 0 à 255 qui est utilisée comme facteur d'échelle pour le résultat interne de la mesure qui est sur 16 bits. Ce résultat est multiplié par Echelle / 256. Ainsi une échelle de 128 diminue-t- elle l'amplitude totale de moitié, une de 64 du quart et ainsi de suite. La commande Alt-P de l'éditeur (voir ci-après) permet de trouver la meilleure échelle possible.

- Variable est utilisée pour stocker le résultat final de l'opération après application de l'échelle. En interne, l'instruction POT fait calculer une valeur sur 16 bits. Cette valeur doit être ramenée à une valeur sur 8 bits grâce au facteur d'échelle. Ce dernier varie en fonction de la plage de variation de la résistance à mesurer.

PAGE 47

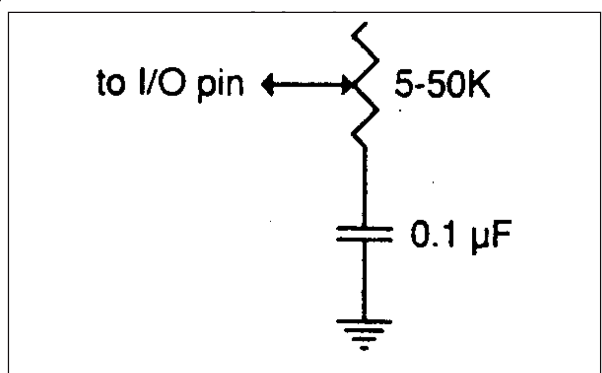
Pour trouver la meilleure valeur de ce facteur d'échelle, l'éditeur dispose de la commande Alt-P. Pour l'utiliser, il faut procéder de la façon suivante :

- Veillez à ce que le Stamp soit connecté au PC et que la résistance en question soit reliée à une de ses pattes.
- Frappez Alt-P pendant que vous êtes sous le contrôle de l'éditeur.
- Une fenêtre s'affiche vous demandant le numéro de la patte à laquelle est connectée la résistance. Répondez par la valeur correspondante comprise entre 0 et 7.
- L'éditeur télécharge alors un court programme dans le Stamp. Notez que ce programme écrase tout programme éventuellement contenu dans l'EEPROM.
- Une nouvelle fenêtre s'ouvre alors indiquant deux valeurs repérées "scale" et "value". Ajustez alors la résistance jusqu'à voir le plus petit nombre possible pour "scale". Lorsque c'est fait, cette valeur doit être prise comme facteur d'échelle et remplace donc échelle dans la syntaxe de l'instruction.
- A titre de vérification optionnelle, vous pouvez vous assurer que le choix réalisé est judicieux en frappant la barre d'espace. Ceci va verrouiller ce facteur d'échelle et ordonner au Stamp de continuer à lire la résistance dont la valeur sera affichée en face de "value" dans la fenêtre. Si vous faites varier cette résistance d'un extrême à l'autre, "value" doit alors évoluer de 0 à 255 à quelques pour cent près. Si vous voulez modifier le facteur d'échelle parce que le résultat n'est pas satisfaisant, frappez à nouveau la barre d'espace ce qui vous ramène à l'étape précédente.

PAGE 48

Exemple :

```
abc:    pot 0,1 00,b2              'lit le potentiomètre relié à la patte 0
        serout 1,n300,(b2)         'envoie sa valeur sous forme série
        goto abc
```



PAGE 49

PULSIN

PULSIN patte, état, variable

Mesure la durée d'une impulsion d'entrée avec une résolution de 10µs.

- Patte est une constante ou variable de 0 à 7 spécifiant le numéro de patte utilisée.

- Etat est une constante ou variable égale à 0 ou 1 qui indique quel front doit avoir lieu avant de commencer la mesure.

- Variable est utilisée pour stocker le résultat de la mesure compris entre 1 et 65536. La variable peut être de type octet ou mot mais, si elle est de type octet et que le résultat de la mesure est supérieur à 2560 µs, il sera mis à zéro.

Exemple :

```
        pulsin 4,1, w2      'mesure la durée d'une impulsion appliquée patte 4
                            'commence la mesure sur une transition montante (1)
                            'termine la mesure à la transition descendante qui suit
                            'place le résultat dans w2
```

PAGE 50

PULSOUT

PULSOUT patte, temps

Génère une impulsion en inversant le niveau d'une patte pendant le laps de temps spécifié.

- Patte est une constante ou variable de 0 à 7 spécifiant le numéro de patte utilisée.

- Temps est une constante ou variable de 0 à 65535 qui spécifie la durée de l'impulsion par pas de 10µs.

Exemple :

```
abc :   pulsout 0,3         'génère une impulsion de 30µs sur la patte 0
        pause 1             'pause de 1 ms
        goto abc            'on répète le processus
```

PAGE 51

PWM

PWM patte, rapport cyclique, cycles

Génère un signal impulsionnel modulé en durée sur la patte spécifiée et place ensuite celle-ci en entrée. Cette instruction peut être utilisée pour produire très facilement une tension analogique grâce à une résistance et un condensateur comme indiqué sur le schéma ci-après. Compte tenu de la décharge inévitable du condensateur au fil du temps, l'instruction doit être répétée régulièrement pour mettre à jour ou rafraîchir cette charge.

- Patte est une constante ou variable de 0 à 7 spécifiant le numéro de patte utilisée.

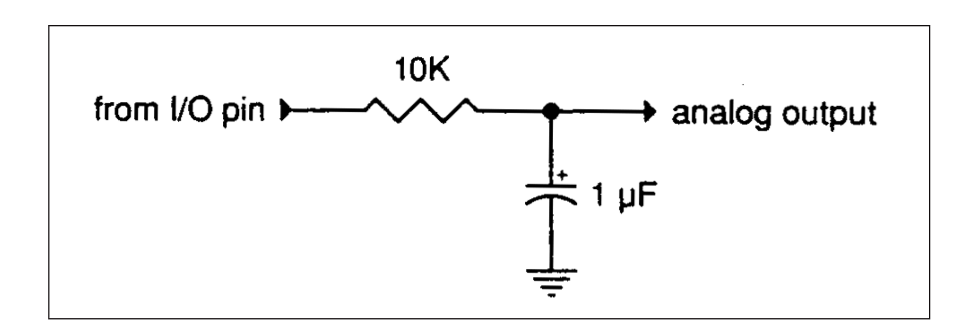
- Rapport cyclique est une constante ou variable de 0 à 255 qui précise le niveau analogique désiré de 0 à 5 volts.

- Cycles est une constante ou variable qui spécifie le nombre de périodes du signal à générer. Chaque cycle prend environ 5 ms.

Nota : toute consommation de courant éventuelle sur le condensateur a pour effet de décharger prématurément celui-ci. Il est donc conseillé de le faire suivre par un montage à haute impédance tel qu'un suiveur à amplificateur opérationnel par exemple.

Exemple :

```
abc :  serin 0,n300,b2     'reçoit une donnée par liaison série
       pwm 1,b2,20         'produit une tension analogique proportionnelle à la valeur reçue.
```



PAGE 52

RANDOM

RANDOM variable

Génère le nombre pseudo aléatoire suivant de la séquence et place celui-ci dans variable.
Le Stamp utilise une suite de 65536 nombres pseudo aléatoires pour exécuter cette instruction.

Lors de cette exécution, la valeur initialement contenue dans variable est utilisée pour choisir un de ces nombres. De ce fait, si la même valeur de départ est utilisée à chaque fois, la même séquence de nombres pseudo aléatoires sera générée.

Pour obtenir une séquence vraiment aléatoire, il faut ajouter un élément incertain au processus. Par exemple vous pouvez lire une entrée série ou l'horloge temps réel et vous en servir comme valeur initiale pour cette instruction.

- Variable est utilisée à la fois comme variable de travail et pour récupérer le nombre pseudo aléatoire généré par l'instruction. Attention cette variable est nécessairement de type mot de 16 bits.

Exemple ..

```
boucle : random w1          'génère un nombre aléatoire dans w1
         sound 1,(b2, 10)   'produit un son utilisant les poids faibles (b2)
                            'du nombre généré pour sélectionner la note
         goto boucle        'on répète le processus
```

PAGE 53

READ

READ adresse, variable

Lit la mémoire EEPROM à l'adresse précisée et place la valeur lue dans variable.
Comme l'EEPROM est utilisée à la fois par votre programme et pas vos données il est prudent de s'assurer que ces dernières ne sont pas écrasées par le programme. Le programme est rangé dans la mémoire en descendant et les données en montant. Pour savoir quoi faire, lisez l'adresse 255 de la mémoire. Elle contient l'adresse de la dernière instruction de votre programme. Vous pouvez donc utiliser librement les adresses inférieures pour vos données.

- **Adresse** est une constante ou variable de 0 à 255 qui spécifie l'adresse à lire.
- **Variable** reçoit la valeur lue dans la mémoire EEPROM.

Exemple :

```
        read 255, b2            'place dans b2 l'adresse de la dernière instruction
                                'de votre programme
```

PAGE 54

RETURN

RETURN

Retour de sous programme. Fait poursuivre l'exécution du programme à l'instruction qui suit immédiatement le GOSUB ayant appelé le sous programme se terminant par cette instruction.
RETURN n'utilise aucun paramètre.

Exemple :

```
        for b4=0 to10
        gosub abc               'appel du sous programme abc
        next
abc :   pulsout 0,b4            'génère une impulsion de b4x10µs sur la patte 0
        toggle 1                'change l'état de la patte 1
        return                  'retourne au programme appelant
```

PAGE 55

REVERSE

REVERSE patte

Inverse le sens de fonctionnement de la patte spécifiée. Si elle était entrée elle devient sortie et vice versa.

- **Patte** est une constante ou variable de 0 à 7 spécifiant le numéro de patte utilisée.

Exemple :

```
        dir3 = 0                'met la patte 3 en entrée
        reverse 3               'met la patte 3 en sortie
        reverse 3               'met à nouveau la patte 3 en entrée
```

PAGE 56

SERIN
SERIN patte, mode, (qualifieur, qualifieur, ...)
SERIN patte, mode, {#}variable, {#}variable, ...
SERIN patte, mode, (qualifieur, qualifieur, ...), {#}variable, {#}variable, ...

Initialise une liaison série sur l'une des pattes du Stamp, configure cette liaison et attend un ou plusieurs qualifieurs optionnels avant de prendre en compte les données reçues. Si les qualifieurs sont utilisés, ils doivent être reçus exactement dans l'ordre spécifié afin que l'exécution de l'instruction puisse continuer. Si des variables sont indiquées, elles se verront affecter les données successives reçues.

Les noms de variables peuvent être précédés d'un symbole dièse (#) optionnel. Dans ce cas, seuls les nombres formés de chiffres de 0 à 9 codés en ASCII seront acceptés pour ces variables. Dans le cas contraire chaque octet reçu est affecté à une variable. Par nombre il faut entendre une suite de chiffres codés en ASCII suivie d'un caractère ASCII non numérique.

Cette possibilité est intéressante pour lire les données numériques émises par de nombreux équipements ne connaissant que le format ASCII. Ainsi par exemple un tel équipement qui voudrait fournir la valeur 108 l'enverrait sous la forme "1", "0" et "8". Avec le symbole dièse cette suite de trois chiffres ASCII pourra être lue sans difficulté.

- Patte est une constante ou variable de 0 à 7 spécifiant le numéro de patte utilisée.

- Mode est une constante ou variable de 0 à 7 spécifiant le mode de fonctionnement de la liaison série conformément aux indications ci-dessous. Mode peut être entrée sous forme numérique de 0 à 7 ou sous la forme symbolique :

N°	Symbole	Vitesse	Mode de codage des données
0	T2400	2400	Vrai
1	T1-200	1200	Vrai
2	T600	600	Vrai
3	T300	300	Vrai
4	N2400	2400	Inversé
5	N1200	1200	Inversé
6	N600	600	Inversé
7	N300	300	Inversé

PAGE 57

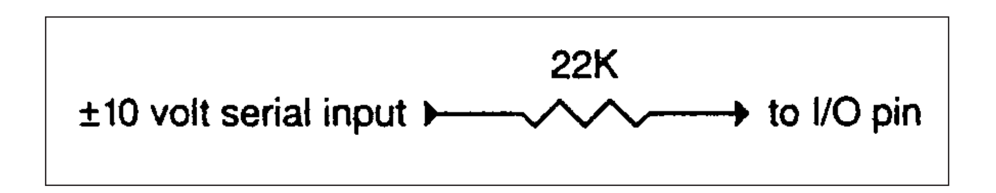
- Qualifieur sont des constantes ou variables optionnelles de 0 à 255 qui doivent être reçus dans l'ordre exact où ils sont spécifiés avant que l'exécution de l'instruction et donc du programme puisse continuer.

- Variables sont utilisées pour recevoir les données arrivant sur la liaison série. Si des qualifieurs ont été spécifiés, ils doivent avoir été reçus dans l'ordre précisé avant que les variables ne puissent se voir affecter les données. Attention! Lors de l'utilisation d'une liaison série sur le Stamp, seuls la vitesse et le mode de codage (vrai ou inversé) des données peuvent être programmés. Le format doit par contre toujours être le suivant: 8 bits de données, pas de parité, 1 bit d'arrêt.

Exemple :

serin 0,N300,("ABC"),b3	Programme la patte 0 comme port série fonctionnant à 300 bauds et attend des données codées en inversé. Après réception des trois caractères ASCII A, B et C dans cet ordre, la donnée suivante est placée dans la variable b3.

Si vous souhaitez que le Stamp puisse recevoir directement des signaux à la norme RS 232, il est possible d'y parvenir avec une simple résistance de 22 kohms comme indiqué ci-après. Les diodes de limitation internes au niveau des pattes des ports assurent en effet la limitation des signaux RS 232 à la plage 0 - 5 volts. Dans ces conditions il faut faire travailler l'instruction en mode N puisque les signaux logiques sur une liaison RS 232 sont inversés.



PAGE 58

SEROUT

SEROUT patte, mode, "#}donnée, (#}donnée, ...)

Initialise une liaison série sur l'une des pattes du Stamp, configure cette liaison et envoie une ou plusieurs données dans l'ordre spécifié.

Les données peuvent être précédées d'un symbole dièse (#) optionnel. Dans ce cas, seuls les nombres formés de chiffres de 0 à 9 codés en ASCII seront acceptés pour ces données. Dans le cas contraire chaque octet de donnée est émis individuellement. Par nombre il faut entendre une suite de chiffres codés en ASCII suivie d'un caractère ASCII non numérique.

Cette possibilité est intéressante pour envoyer les données numériques émises à un équipement ne connaissant que le format ASCII. Ainsi par exemple pour envoyer 108 à un tel équipement il faut l'envoyer sous la "1", "0" et "8". Avec le symbole dièse cela ne présentera aucune difficulté, il suffira d'écrire #108.

- **Patte** est une constante ou variable de 0 à 7 spécifiant le numéro de patte utilisée.

- **Mode** est une constante ou variable de 0 à 15 spécifiant le mode de fonctionnement de la liaison série conformément aux indications ci-dessous. Mode peut être entrée sous forme numérique de 0 à 15 ou sous la forme symbolique :

N°	Symbole	Vitesse	Codage des données	Type de sortie
0	T2400	2400	Vrai	Active
1	T1200	1200	Vrai	Active
2	T600	600	Vrai	Active
3	T300	300	Vrai	Active
4	N2400	2400	Inversé	Active
5	N1200	1200	Inversé	Active
6	N600	600	Inversé	Active
7	N300	300	Inversé	Active
8	OT2400	2400	Vrai	Drain ouvert
9	OT1200	1200	Vrai	Drain ouvert
10	OT600	600	Vrai	Drain ouvert
11	OT300	300	Vrai	Drain ouvert
12	ON2400	2400	Inversé	Source ouverte
13	ON1200	1200	Inversé	Source ouverte
14	ON600	600	Inversé	Source ouverte
15	ON300	300	Inversé	Source ouverte

PAGE 59

- Donnée est une constante ou variable de 0 à 255 qui est émise sur la liaison série.

Attention!

Lors de l'utilisation d'une liaison série sur le Stamp, seuls la vitesse et le mode de codage (vrai ou inversé) des données peuvent être programmés. Le format par contre est toujours le suivant : 8 bits de données, pas de parité, 1 bit d'arrêt.

Exemple :

serout 1,N300, (b2)   Programme la patte 1 comme port série fonctionnant à 300 bauds et envoie la donnée contenue dans b2 codée en inversé.

Attention!

Le Stamp ne sait produire que des niveaux logiques TTL, dans la plage 0 - 5 volts donc. Des circuits d'interface peuvent donc être nécessaires pour travailler sur des liaisons séries RS 232.

PAGE 60

SLEEP

SLEEP secondes

Met le Stamp en mode sommeil pendant un certain nombre de secondes. La résolution de cette instruction est de une seconde et sa précision de l'ordre de 99,9%. La consommation du Stamp en mode sommeil est de 20µA hors entrées/sorties éventuelles.

- Secondes est une constante ou variable de 0 à 65535 qui spécifie-la durée du mode sommeil. Elle peut donc aller de une seconde à environ 18 heures.

Exemple :

sleep 3600   'met le Stamp en mode sommeil pour une heure environ

PAGE 61

SOUND

SOUND patte, (note, durée, note, durée, ...)

Joue les notes spécifiées avec les durées spécifiées. La patte de sortie utilisée doit être connectée à la charge par un condensateur dont le pole positif est relié à cette patte comme indiqué sur la figure ci-après. Un haut-parleur haute impédance (40 ohms) ou un buzzer piézo peut être utilisé. Les sons sont produits au moyen d'un signal carré.

- Patte est une constante ou variable de 0 à 7 spécifiant le numéro de patte utilisée.

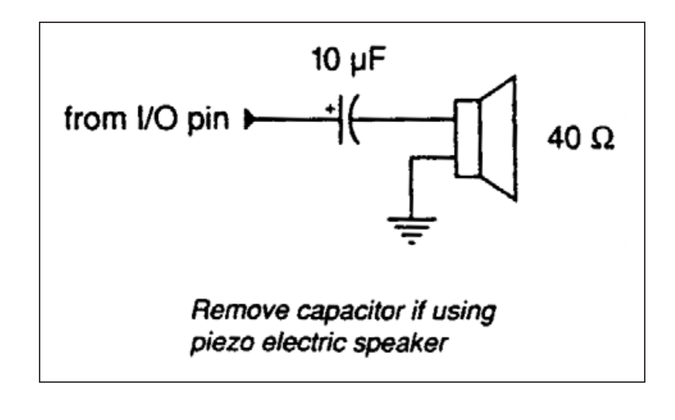
- Note est une constante ou variable de 0 à 255 qui spécifie le type et la fréquence de la note. Un 0 produit un silence, des notes ascendantes sont produites de 1 à 127 et des bruits blancs ascendants sont produits de 128 à 255.

- Durée est une constante ou variable de 0 à 255 qui fixe la durée de chaque note.

Exemple :

for b2 = 0 to 255

sound 1, (25, 10, b2, 10)   'génère un note fixe suivie d'une note ascendante , les deux ayant la même durée



PAGE 62

TOGGLE

TOGGLE patte

Met la patte spécifiée en sortie si elle ne l'était pas déjà et change son état (haut vers bas ou vice versa).

- **Patte** est une constante ou variable de 0 à 7 spécifiant le numéro de patte utilisée.

Exemple :

```
for b2 = 1 to 25
toggle 5                'change l'état de la patte 5
next
```

PAGE 63

WRITE

WRITE adresse, donnée

Ecrit la donnée à l'adresse spécifiée dans la mémoire EEPROM.

Comme l'EEPROM est utilisée à la fois par votre programme et pas vos données il est prudent de s'assurer que ces dernières ne sont pas écrasées par le programme. Le programme est rangé dans la mémoire en descendant et les données en montant. Pour savoir quoi faire, lisez l'adresse 255 de la mémoire. Elle contient l'adresse de la dernière instruction de votre programme. Vous pouvez donc utiliser librement les adresses inférieures pour vos données.

- **Adresse** est une constante ou variable de 0 à 255 qui spécifie l'adresse où écrire.

- **Donnée** est une constante ou variable de 0 à 255 à écrire dans la mémoire EEPROM.

Exemple :

```
          read 255,b2       'lit l'adresse de la dernière instruction du programme
boucle :  b2 = b2 - 1       'sélectionne la première adresse disponible
          serin O,N300,b3   'reçoit une donnée série dans b3
          write b2,b3       'écrit la donnée reçue à l'adresse contenue dans b2
          if b2 > 0 then boucle
```

Produit importé et distribué par :

Selectronic

86 rue de CAMBRAI

59000 LILLE

TEL : 0 328 550 328   Fax : 0 328 550 329

SAV : 0 328 550 323   www.selectronic.fr

03/2003 /MZ